Diese Website verwendet Cookies von Google, um Dienste anzubieten und den Traffic zu analysieren. Deine IP-Adresse und dein User-Agent werden zusammen mit Messwerten zur Leistung und Sicherheit für Google freigegeben. So können Nutzungsstatistiken generiert, Missbrauchsfälle erkannt und behoben und die Qualität des Dienstes gewährleistet werden.

WEITERE INFORMATIONEN OK

An unending variety of topics

Saturday, December 10, 2011

Short takes

During my blog absence, I've been studying, designing, and implementing a style of programming I call *temporal programming*. It is useful for, among other things, implementing smart contracts. Meanwhile, I encourage readers interested in programming to check out Node.js. Temporal programming starts with event-oriented programming and takes it further. Temporal programming will give us control over *when* our instructions get executed: the plodding do this, then that, then the other, as if machine activities are only supposed to happen in one big long sequence and merely output some big long tape, will be relegated to secondary status. More to come.

Is Netflix management really stupid? To summarize Megan McCardle, no: under copyright law, DVDs are covered by the first sale rule -- once you buy a DVD, you can't make a copy, but you can sell, rent or give away the DVD itself. Streaming, on the other hand, essentially involves making a copy, and you can't do it legally without the copyright owner's permission. So to stream the most demanded content, you usually need the permission of owners, and thus have to pay what they demand.

It thus makes complete sense that Netflix has to charge high prices for streaming -- because for the kind of content they want to stream (i.e. the copyrights that are so valuable that their owners bother to prevent the content from being distributed on YouTube), content owners are demanding revenues similar to what they are accustomed to via cable.

I'd add that the first sale rule also explains why Netflix, contrary to its name, originally succeeded in out-competing a gaggle of Internet video streaming companies by the stone-age method of shipping DVDs by mail. The main thing Netflix may have done wrong was, after succeeding in pivoting from Internet streaming to mailing DVDs, going back to their original goal in a way that set false expectations about prices (i.e. that streaming prices would be closer to DVD rental prices than to cable TV). Probably what happened is that Netflix CEO Reed Hastings thought he could use his mail-order rental business as leverage to negotiate lower prices with copyright owners, but this strategy did not succeed.

It also makes sense that Netflix (and their streaming competitors) lack licensed content due to copyright owners' long-standing aversion to Internet streaming. All this was happening to Netflix's video-streaming competitors long before Netflix's much more recent emphasis on that business. Netflix apparently hasn't, after all, solved the institutional problem that their DVD-shipping model worked around.

Pages

- Home

About Me

Nick Szabo

"A premier thinker about history, law and economics, and the lessons they have for security." -- Adam Shostack, Emergent Chaos

"Szabo comes out with these essays that leave me in awe." -- Brian Dunbar

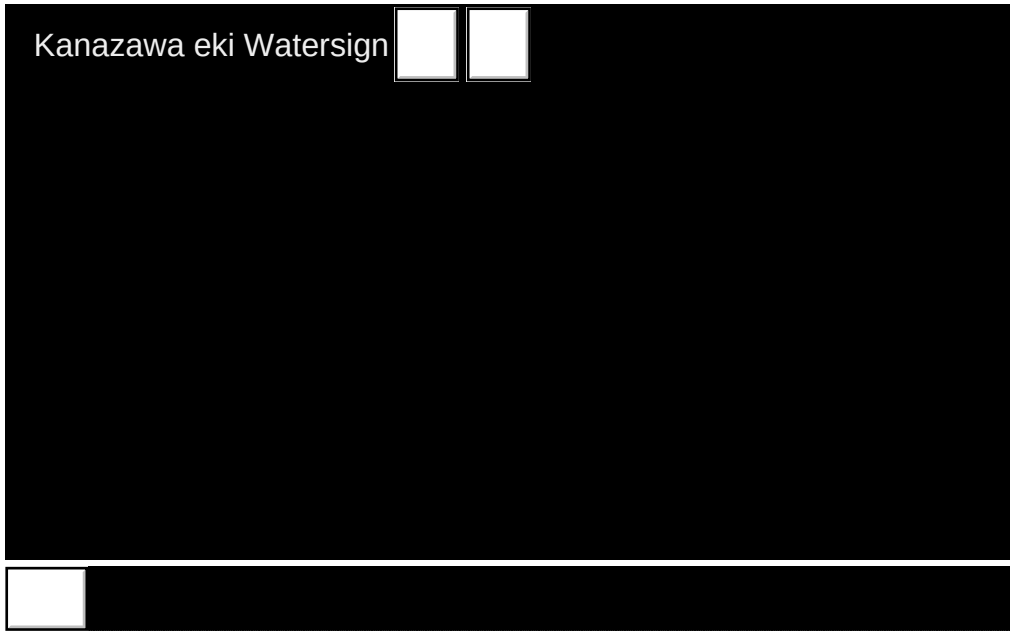
"Reading material that is eclectic, challenging and endlessly fascinating." -- Sean McGrath, Propylon

"Like most blogs worth my attention, this blog is updated only infrequently. That is because the authors of blogs worth my attention only post when they have something to say that is true, relevant and not already known by their audience. Most of the human race does not have the skill to know when an idea has these three properties. The skill is particularly rare in the fields of politics and economics, which is why this blog is such a rare and valuable thing." -- Richard Hollerith

View my complete profile

Bit gold and I make brief appearances in Wired's Bitcoin article.

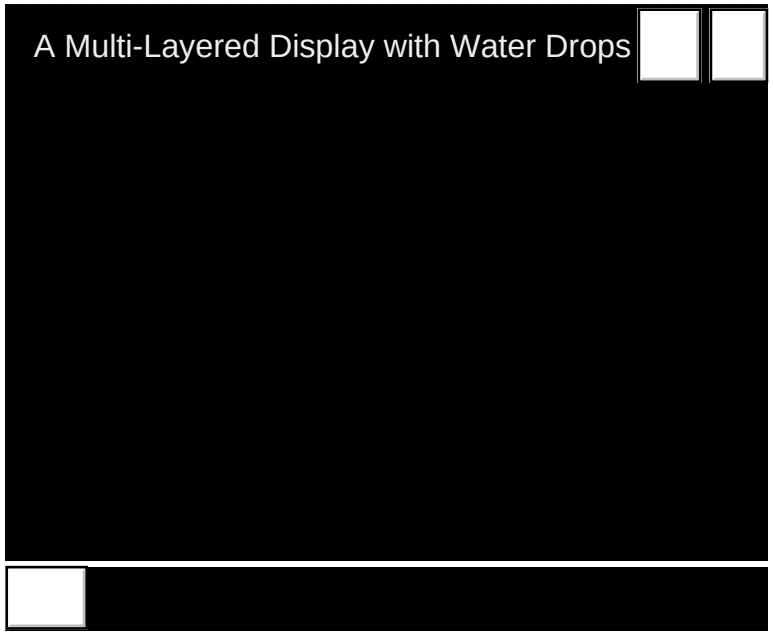
Water remains fun. Digital fountains show how precisely drops can be located and timed:



Blog Archive

- ► 2018 (1)
- ► 2017 (3)
- ► 2016 (4)
- ► 2015 (3)
- ► 2014 (3)
- ► 2013 (3)
- ► 2012 (8)
- ▼ 2011 (12)
  - ▼ December (1)
    - Short takes
  - ► July (1)
  - ► June (2)
  - ► May (2)
  - ► February (2)
  - ► January (4)
- ► 2010 (9)
- ► 2009 (29)
- ► 2008 (55)
- ► 2007 (47)
- ► 2006 (130)
- ► 2005 (44)

This one is a bit different, drops fall with uniform regularity but are used to display light:



Posted by Nick Szabo at 1:07 PM

6 comments:

Jay Dugger said...

Good to read your posts again, N.S.. I eagerly await the next posts.

4:20 AM

Anonymous said...

Yep, nice to see you post again.

3:22 PM

Brian Dunbar said...

I've been studying, designing, and implementing a style of programming I call temporal programming.

I learned - in a sort of very basic way - how to program in the Marines. Never used it there, got shuffled off into small computer systems: peecees, LANs, WANs, etc.

But my job as a system admin is evolving more and more into 'coding': shell scripts of course, python, perl. Configuration engines where one programs the conf engine to deploy software to the host.

Anyway - I've been re-learning how to write code. I'm trying to learn common LISP and that's -fun-.

Seems clear that a guy who can write code and understand systems is the 21st century equiv of a steam engineer in the 18th.

So .. I'd be keen to read more about temporal programming, when you have the time.

7:54 PM

nick said...

Jay and Anonymous, thanks for your kind comments.

Brian, if you've used Unix cron to schedule jobs, that gives you a taste of temporal programming. Besides scheduling specific times when functions should execute (rather than procedural straightjacket of merely executing one function after another, or the functional straightjacket of merely executing one function inside another), the other big part of temporal programming is responding to unpredictable events, which allows us to execute code when the user or unpredictable outside processes want or require us to.

While I have designed my own purely temporal language, there is not yet a sufficiently working implementation, so I suggest the next best thing -- JavaScript. The main reason being, JavaScript as a language for implementing web user interfaces has created a culture full of programmers who understand event-oriented programming, programming where callbacks are the norm rather than the exception. That is most of temporal programming.

JavaScript also has features that make callbacks, and thus event-oriented and temporal programming, much easier to write correctly than in most other languages. With JavaScript one can implement, albeit in a more awkward way, all the temporal constructs that one can in my language. So, the examples I will give will be in JavaScript.

(Another good choice would be Erlang, but the population of programmers, and the practical uses for their talents, are far smaller).

My main motivation for temporal programming has been smart contracts, although there are many other uses. The basic building blocks of financial contracts are the future, the bond, and the option. In essence these are very temporal creatures:

(1) A future is a scheduled swap of one instrument or property for another at a specific future date.

(2) A bond is a schedule of payments (or more generally one can have a schedule of swaps).

(3) An option gives its holder the choice of executing a swap during a certain period of time.

So with financial contracts (and some other instances of bringing legal documents to life on computer), workflows, etc., the basic building blocks are deadlines, periods, schedules, and real-world or user-initiated events. Temporal programming.

10:50 PM

LDiracDelta said...

You might be interested in investigating some prior art. RTL hardware design and verification languages like System Verilog and Specman make extensive use of temporal constructs during simulation and we've been using these constructs for over 15 years. See [1], [2]. Also, you might be interested in the construction of their runtime environments or "simulation kernels" for scheduling events.

10:05 AM

Beyond Anon said...

For pretty much[1] all of the existence of Homo sapiens except for the last two or three hundred years[2] most of us have lived at the Malthusian Margins.

It seems to me that this must have exerted enormous selection pressure on humans, and that most people will have genetic support for understanding and operating in zero-sum environments.

It might be that we are heading back to the Malthusian Margins.

[1] The expansion of some humans into the Americas must have been a time when some humans have been very far from the Malthusian Margins.

[2] Not all humans during these last few hundred years have been far from the Malthusian Margins, as they have not all been able to partake of the marvels of the Industrial Revolution and particularly the new sources of energy and new lands to expand into.

8:25 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Simple theme. Powered by Blogger.